GLAD package: Gain and Loss Analysis of DNA

Philippe Hupe[1,2] and Emmanuel Barillot[2]

May 13, 2009

1. UMR 144 CNRS/Institut Curie, Institut Curie, 26, rue d'Ulm, Paris, 75248 cedex 05, France
2. Service Bioinformatique, Institut Curie, 26, rue d'Ulm, Paris, 75248 cedex 05, France
`glad@curie.fr` `http://bioinfo.curie.fr`

Contents

1 Overview

This document presents an overview of the `GLAD` package (Gain and Loss Analysis of DNA). This package is devoted to the analysis of Array Comparative Genomic Hybridization (array CGH) (Pinkel et al., 1998; Snijders et al., 2001; Solinas-Toldo et al., 1997; Ishkanian et al., 2004). The methodology for detecting the breakpoints delimiting altered regions in genomic patterns and assigning a status (normal, gained or lost) to each chromosomal region described in the paper Hupé et al. (2004) is implemented in this package. Some graphical functions are provided as well.

2 Data

2.1 Public data set

We used the public data set described in Snijders et al. (2001). The data corresponds to 15 human cell strains with known karyotypes (12 fibroblast cell strains, 2 chorionic villus cell strains, 1 lymploblast cell strain) from the NIGMS Human Genetics Cell Repository (`http://locus.umdnj.edu/nigms`). Each cell strain has been hybridized on CGH arrays of 2276 BACs, spotted in triplicates. Two array CGH profiles from the data obtained by Veltman et al. (2003) are available.

2.2 Bladder cancer data

Bladder cancer data from tumors collected at Henri Mondor Hospital (CrÃl'teil, France) (Billerey et al., 2001) have been hybridized on CGH arrays composed of 2464 BACs (Radvanyi, Pinkel et al., unpublished results). In this data, only the log-ratios are provided and no information about clones is available since the data is not yet published. This data allows only some graphical functionalities to be shown and will be used as a support to illustrate some functions for array normalization (not yes available in the current version of the package).

3 GLAD classes

3.1 arrayCGH

This class stores raw values after images analysis. The object arrayCGH is a list with at least a data.frame named arrayValues and a vector named arrayDesign. The data.frame arrayValues must contain the following fields:

Col Vector of columns coordinates.

Row Vector of rows coordinates.

... Other elements can be added.

The vector arrayDesign is composed of 4 values: c(arrayCol, arrayRow, SpotCol, SpotRow). The array CGH is represented by arrayRow*arrayCol blocs and each bloc is composed of SpotRow*SpotCol spots. N.B.: Col takes the values in 1:arrayRow*SpotRow and Row takes the values in 1:array-Col*SpotCol

3.2 profileCGH and profileChr

This class stores synthetic values related to each clone available on the arrayCGH. The object profileChr corresponds to data of only one chromosome. Objects profileCGH and profileChr are composed of a list with the first element profileValues which is a data.frame with the following columns names:

LogRatio Test over Reference log-ratio.

PosOrder The rank position of each clone on the genome.

PosBase The base position of each clone on the genome.

Chromosome Chromosome name.

Clone The name of the corresponding clone.

... Other elements can be added.

LogRatio, Chromosome and PosOrder are compulsory.
To create those objects you can use the function *as.profileCGH*.

4 Analysis of array CGH profile

Two functions are available: *glad* and *daglad*. The second one is an improvment of of first one which was originally describes in Hupé et al. (2004). We recommand to use the *daglad* function. For fast computation use the option *smoothfunc=haarseg*.

4.1 Segmentation algorithms

Two algorithms are available for data segmentation:

- AWS (Polzehl and Spokoiny, 2000, 2002)
- HaarSeg (Ben-Yaacov and Eldar, 2008)

4.2 The *glad* function

A result of the GLAD methodology on cell line gm13330 (Snijders et al., 2001) is presented in **Figure 1**.

```
######################################################################################

Have fun with GLAD

For smoothing it is possible to use either
the AWS algorithm (Polzehl and Spokoiny, 2002)
or the HaarSeg algorithm (Ben-Yaacov and Eldar, Bioinformatics,  2008)

If you use the package with AWS, please cite:
Hupe et al. (Bioinformatics, 2004) and Polzehl and Spokoiny (2002)

If you use the package with HaarSeg, please cite:
Hupe et al. (Bioinformatics, 2004) and (Ben-Yaacov and Eldar, Bioinformatics, 2008)

For fast computation it is recommanded to use
the daglad function with smoothfunc=haarseg

######################################################################################
```

```
> data(snijders)
> profileCGH <- as.profileCGH(gm13330)
> res <- glad(profileCGH, mediancenter = FALSE, smoothfunc = "lawsglad",
+     bandwidth = 10, round = 1.5, model = "Gaussian", lkern = "Exponential",
+     qlambda = 0.999, base = FALSE, lambdabreak = 8, lambdacluster = 8,
+     lambdaclusterGen = 40, type = "tricubic", param = c(d = 6),
+     alpha = 0.001, msize = 5, method = "centroid", nmax = 8,
+     verbose = FALSE)
```

```
[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "subset"
Time difference of 0.0002999306 secs
  Region Card         Var       Mean     VarLike
1      1   82 0.008020255 0.01801656 0.008020255
3      3   46 0.011707465 0.52718028 0.011707465
[1] "aggregation"
Time difference of 0.02336502 secs
[1] "method"
[1] 7
[1] "hclust"
Time difference of 0.0009338856 secs
[1] "cluster"
Time difference of 0.009972095 secs
[1] "VERIF"
integer(0)
[1] "END VERIF"
[1] "merge"
Time difference of 0.001362085 secs
[1] "clustering"
Time difference of 0.0004189014 secs
[1] "Temps findCluster: 0.0363519191741943"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.982948e-05 secs
[1] "Temps findCluster: 0.000239849090576172"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000217914581298828"
```

```
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "subset"
Time difference of 0.0002908707 secs
  Region Card         Var       Mean     VarLike
6      6  150 0.009330993 -0.0686637 0.009330993
7      7   17 0.004037443 -0.8388732 0.004037443
[1] "aggregation"
Time difference of 0.004235029 secs
[1] "method"
[1] 7
[1] "hclust"
Time difference of 0.0006990433 secs
[1] "cluster"
Time difference of 0.00899601 secs
[1] "VERIF"
integer(0)
[1] "END VERIF"
[1] "merge"
Time difference of 0.001302958 secs
[1] "clustering"
Time difference of 0.0003941059 secs
[1] "Temps findCluster: 0.0159180164337158"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.482269e-05 secs
[1] "Temps findCluster: 0.000232934951782227"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000224828720092773"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000238895416259766"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
```

```
[1] "clustering"
Time difference of 4.506111e-05 secs
[1] "Temps findCluster: 0.000221967697143555"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.386902e-05 secs
[1] "Temps findCluster: 0.000213861465454102"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.506111e-05 secs
[1] "Temps findCluster: 0.000226020812988281"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000240802764892578"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.386902e-05 secs
[1] "Temps findCluster: 0.000209093093872070"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000210046768188477"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.506111e-05 secs
[1] "Temps findCluster: 0.000206232070922852"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
```

```
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000221014022827148"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 5.483627e-05 secs
[1] "Temps findCluster: 0.000233888626098633"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000221014022827148"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000208139419555664"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000207901000976562"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.506111e-05 secs
[1] "Temps findCluster: 0.00020599365234375"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.506111e-05 secs
[1] "Temps findCluster: 0.000204086303710938"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.506111e-05 secs
```

```
[1] "Temps findCluster: 0.000204086303710938"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.696846e-05 secs
[1] "Temps findCluster: 0.000213861465454102"
[1] "centroid"
findCluster.profileChr(profileChr = profileCGH, region = "ZoneChr",
    genome = TRUE, lambda = lambdaclusterGen, nmin = 1, nmax = nmax,
    type = type, param = param, verbose = verbose, method = method)
[1] "subset"
Time difference of 0.0006768703 secs
   Region Card         Var         Mean     VarLike
1       1   82 0.008020255  0.018016561 0.008020255
2       2   46 0.011707465  0.527180283 0.011707465
3       3   65 0.008591739 -0.019321185 0.008591739
4       4   83 0.006330242 -0.043528434 0.006330242
5       5  150 0.009330993 -0.068663693 0.009330993
6       6   17 0.004037443 -0.838873176 0.004037443
7       7   97 0.008216950 -0.015118722 0.008216950
8       8   83 0.010461284 -0.009310735 0.010461284
9       9  174 0.008416583  0.018234207 0.008416583
10     10  151 0.007494192 -0.030153377 0.007494192
11     11  107 0.010148660 -0.032047607 0.010148660
12     12  127 0.010891078  0.014396394 0.010891078
13     13  180 0.009721379 -0.011066117 0.009721379
14     14   88 0.006625335  0.007699148 0.006625335
15     15   47 0.005636744 -0.045899702 0.005636744
16     16   71 0.011104201 -0.008812183 0.011104201
17     17   65 0.009007422  0.010588015 0.009007422
18     18   64 0.007821951  0.008468625 0.007821951
19     19   86 0.006993832  0.022920081 0.006993832
20     20   50 0.012088798  0.019128380 0.012088798
21     21   37 0.007851133  0.069814027 0.007851133
22     22   87 0.007511773  0.036766563 0.007511773
23     23   32 0.010182556 -0.074258406 0.010182556
24     24   15 0.006725160  0.056583867 0.006725160
25     25   54 0.004065299 -0.055199741 0.004065299
[1] "aggregation"
Time difference of 0.007138014 secs
[1] "method"
[1] 7
[1] "hclust"
Time difference of 0.0007610321 secs
[1] "cluster"
Time difference of 0.06416392 secs
[1] "VERIF"
integer(0)
```

```
[1] "END VERIF"
[1] "merge"
Time difference of 0.002378225 secs
[1] "clustering"
Time difference of 0.0004589558 secs
[1] "Temps findCluster: 0.0755770206451416"
[1] "Results Preparation"
```

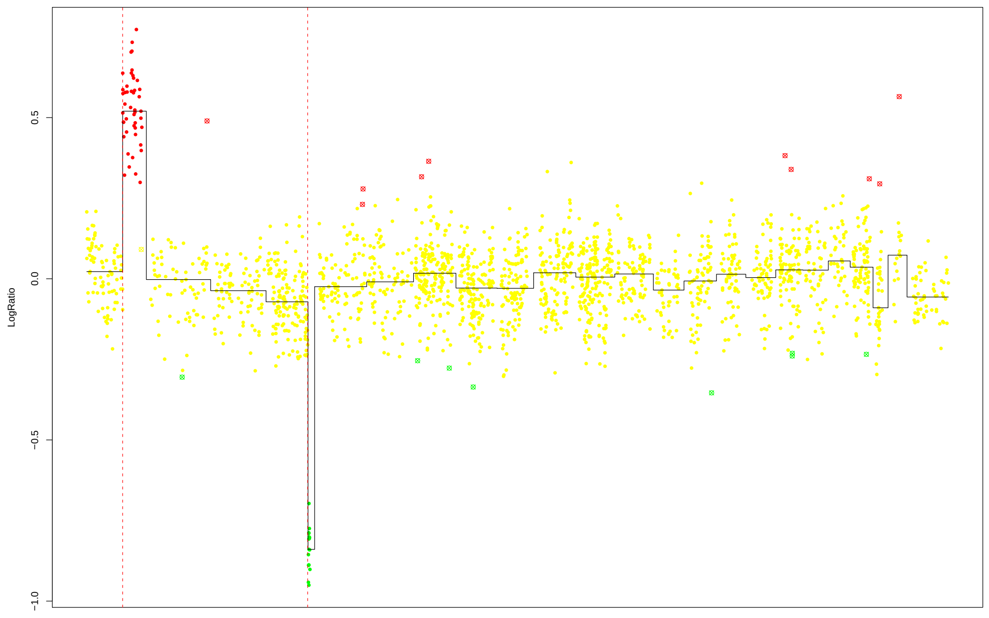

Figure 1: Results of glad on cell line gm13330 (Snijders data).

4.3 The *daglad* function

The algorithm implemented in this function is a slightly modified version of the GLAD algorithm.

```
> data(veltman)
> profileCGH <- as.profileCGH(P9)
> profileCGH <- daglad(profileCGH, mediancenter = FALSE, normalrefcenter = FALSE,
+     genomestep = FALSE, smoothfunc = "lawsglad", lkern = "Exponential",
+     model = "Gaussian", qlambda = 0.999, bandwidth = 10, base = FALSE,
+     round = 1.5, lambdabreak = 8, lambdaclusterGen = 40, param = c(d = 6),
+     alpha = 0.001, msize = 5, method = "centroid", nmin = 1,
+     nmax = 8, amplicon = 1, deletion = -5, deltaN = 0.2, forceGL = c(-0.3,
+         0.3), nbsigma = 3, MinBkpWeight = 0.35, CheckBkpPos = TRUE)
```

```
[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints"
[1] "DNA copy number calling"
[1] "centroid"
findCluster.profileChr(profileChr = profileCGH, region = "NormalRange",
    genome = TRUE, lambda = lambdaclusterGen, nmin = nmin, nmax = nmax,
    verbose = verbose, method = method)
[1] "subset"
Time difference of 0.0004310608 secs
   Region Card         Var        Mean     VarLike
0       0 1073 0.020911942  0.02387856 0.020911942
3       3    9 0.016106667 -1.05237556 0.016106667
5       5    6 0.011990977  0.40234667 0.011990977
16     16  102 0.018418069 -0.68389794 0.018418069
19     19   84 0.018317330 -0.23673369 0.018317330
20     20   51 0.013487584  0.28811588 0.013487584
21     21   67 0.014696131 -0.60601299 0.014696131
23     23   20 0.015848997 -0.62260500 0.015848997
24     24   28 0.008817172  0.22632679 0.008817172
30     30   19 0.009314540 -0.73866579 0.009314540
35     35   20 0.032597530  0.19617450 0.032597530
36     36   13 0.016287276 -0.65304308 0.016287276
39     39   54 0.010813561  0.34427130 0.010813561
40     40   18 0.014156788  0.32048333 0.014156788
41     41   13 0.032776008 -0.77201692 0.032776008
[1] "aggregation"
Time difference of 0.005933046 secs
[1] "method"
[1] 7
[1] "hclust"
Time difference of 0.0008471012 secs
[1] "cluster"
Time difference of 0.06051993 secs
[1] "VERIF"
integer(0)
[1] "END VERIF"
[1] "merge"
```

```
Time difference of 0.003655910 secs
[1] "clustering"
Time difference of 0.0004730225 secs
[1] "Temps findCluster: 0.0718600749969482"
[1] "jointure BkpInfo"
   user  system elapsed
  0.000   0.000   0.002
[1] "Check Breakpoints Position"
[1] "Results Preparation"
```

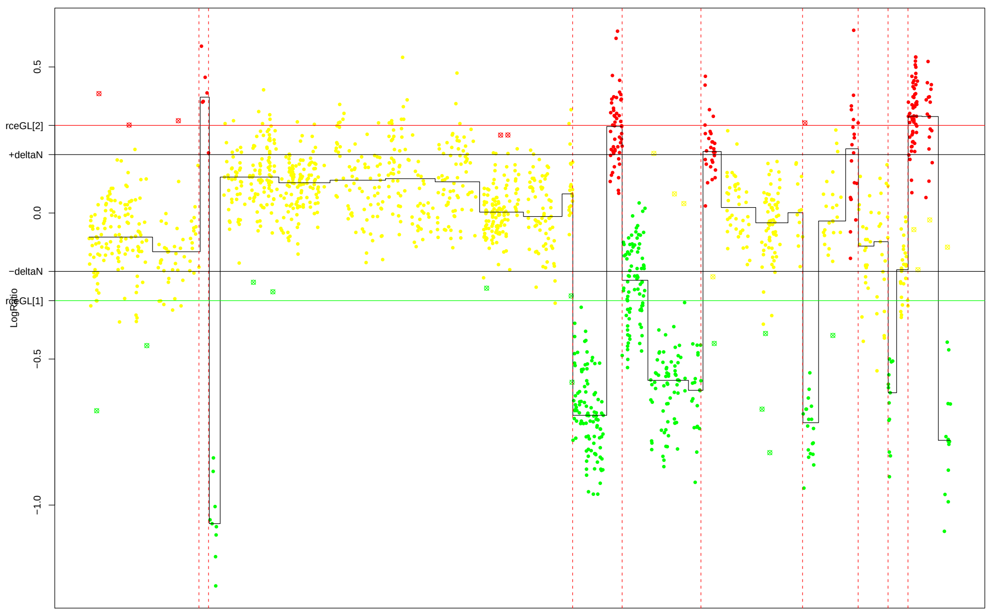

Figure 2: Results of daglad on the patient P9 (Veltman data).

The *daglad* function allows to choose some threshold to help the algorithm to identify the status of the genomic regions. The thresholds are given in the following parameters:

- deltaN
- forceGL
- deletion
- amplicon

Comparing **Figure 2** and **Figure 3** shows the influence of two different sets of parameters.

```
> data(veltman)
> profileCGH <- as.profileCGH(P9)
> profileCGH <- daglad(profileCGH, mediancenter = FALSE, normalrefcenter = FALSE,
+     genomestep = FALSE, smoothfunc = "lawsglad", lkern = "Exponential",
+     model = "Gaussian", qlambda = 0.999, bandwidth = 10, base = FALSE,
+      round = 1.5, lambdabreak = 8, lambdaclusterGen = 40, param = c(d = 6),
+      alpha = 0.001, msize = 5, method = "centroid", nmin = 1,
+      nmax = 8, amplicon = 1, deletion = -5, deltaN = 0.1, forceGL = c(-0.15,
+          0.15), nbsigma = 3, MinBkpWeight = 0.35, CheckBkpPos = TRUE)

[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints"
[1] "DNA copy number calling"
[1] "centroid"
findCluster.profileChr(profileChr = profileCGH, region = "NormalRange",
    genome = TRUE, lambda = lambdaclusterGen, nmin = nmin, nmax = nmax,
    verbose = verbose, method = method)
[1] "subset"
Time difference of 0.000428915 secs
   Region Card         Var        Mean     VarLike
0       0  507 0.014952197 -0.02392314 0.014952197
3       3    9 0.016106667 -1.05237556 0.016106667
4       4   47 0.011014955 -0.13597277 0.011014955
5       5    6 0.011990977  0.40234667 0.011990977
6       6  131 0.010552460  0.12753221 0.010552460
7       7  119 0.007360982  0.09284193 0.007360982
8       8   76 0.015480130  0.10562566 0.015480130
10     10   72 0.018256433  0.11874667 0.018256433
12     12   72 0.015374382  0.11053208 0.015374382
16     16  102 0.018418069 -0.68389794 0.018418069
19     19   84 0.018317330 -0.23673369 0.018317330
20     20   51 0.013487584  0.28811588 0.013487584
21     21   67 0.014696131 -0.60601299 0.014696131
23     23   20 0.015848997 -0.62260500 0.015848997
24     24   28 0.008817172  0.22632679 0.008817172
30     30   19 0.009314540 -0.73866579 0.009314540
33     33   22 0.016982902 -0.12474818 0.016982902
35     35   20 0.032597530  0.19617450 0.032597530
36     36   13 0.016287276 -0.65304308 0.016287276
38     38   27 0.008868570 -0.20017111 0.008868570
39     39   54 0.010813561  0.34427130 0.010813561
40     40   18 0.014156788  0.32048333 0.014156788
41     41   13 0.032776008 -0.77201692 0.032776008
[1] "aggregation"
Time difference of 0.006886959 secs
[1] "method"
[1] 7
[1] "hclust"
Time difference of 0.0007531643 secs
[1] "cluster"
Time difference of 0.05903792 secs
[1] "VERIF"
integer(0)
[1] "END VERIF"
[1] "merge"
```

```
Time difference of 0.002163887 secs
[1] "clustering"
Time difference of 0.0004131794 secs
[1] "Temps findCluster: 0.0696840286254883"
[1] "jointure BkpInfo"
   user  system elapsed
  0.000   0.000   0.002
[1] "Check Breakpoints Position"
[1] "Results Preparation"
```

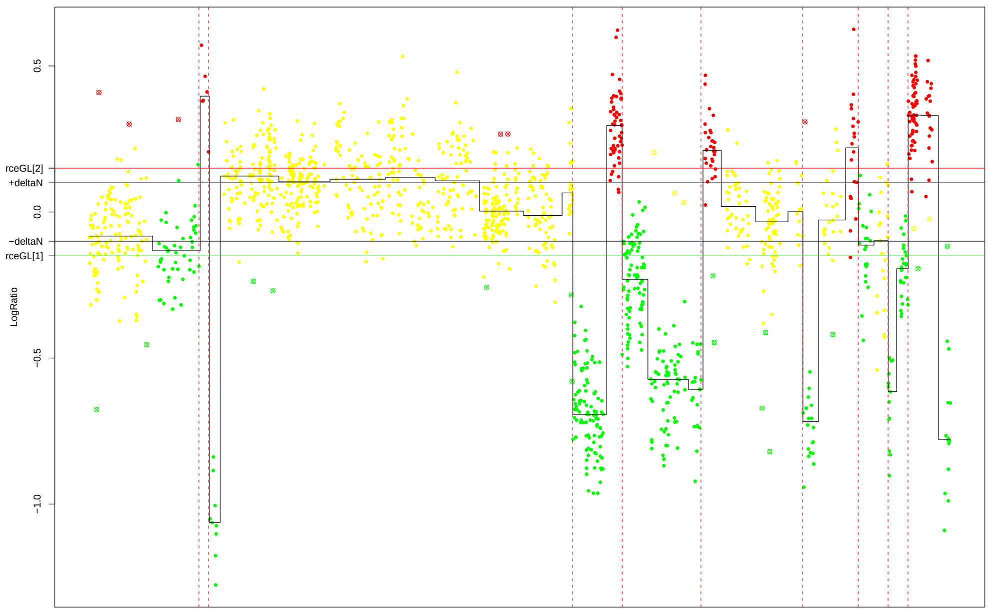

Figure 3: Results of daglad on the patient P9 (Veltman data) - Influence of the thresholds.

The *daglad* function allows a smoothing step over the whole genome (if *genomestep=TRUE*) where all the chromosomes are concatenated together. During this step, the cluster which corresponds to the Normal DNA level is identified: the thresholds used in the function (deltaN, forceGL, amplicon, deletion) are then compared to the median of this cluster.

4.4 Tuning parameters

The most important parameters are:

- *lambdabreak*
- *lambdacluster*
- *lambdaclusterGen*
- *param* $c(d = 6)$

Decreasing those parameters will lead to a higher number of breakpoints identified. For arrays experiments with very small Signal to Noise ratio it is recommended to use a small value of *param* like $d = 3$ or less.

5 Graphical functions

5.1 Plot of raw array data

```
> data(arrayCGH)
> array <- list(arrayValues = array2, arrayDesign = c(4, 4, 21,
+     22))
> class(array) <- "arrayCGH"
> arrayPlot(array, "Log2Rat", bar = "none")
```

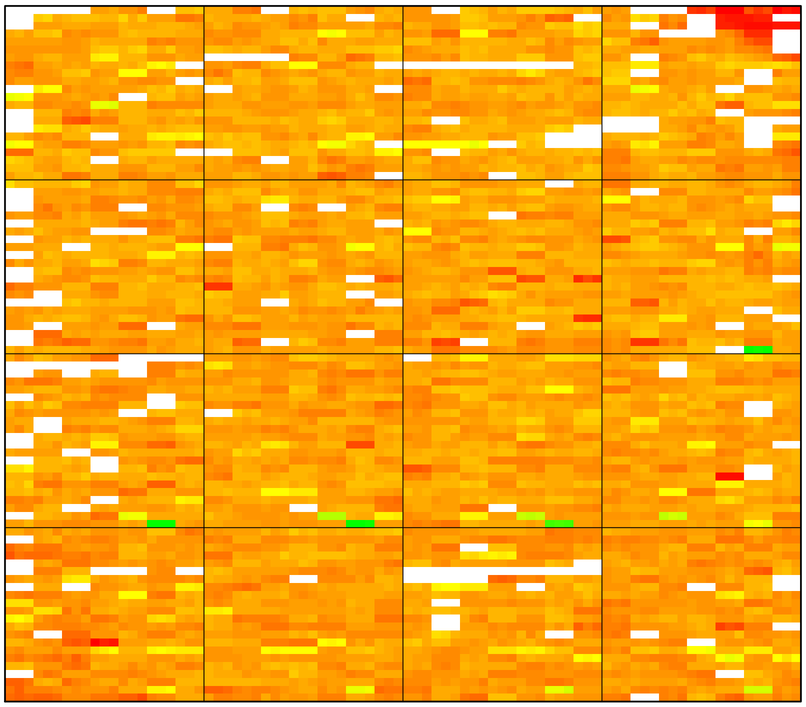

Figure 4: Spatial image of array CGH

```
> arrayPersp(array, "Log2Rat", box = FALSE, theta = 110, phi = 40,
+     bar = FALSE)
```

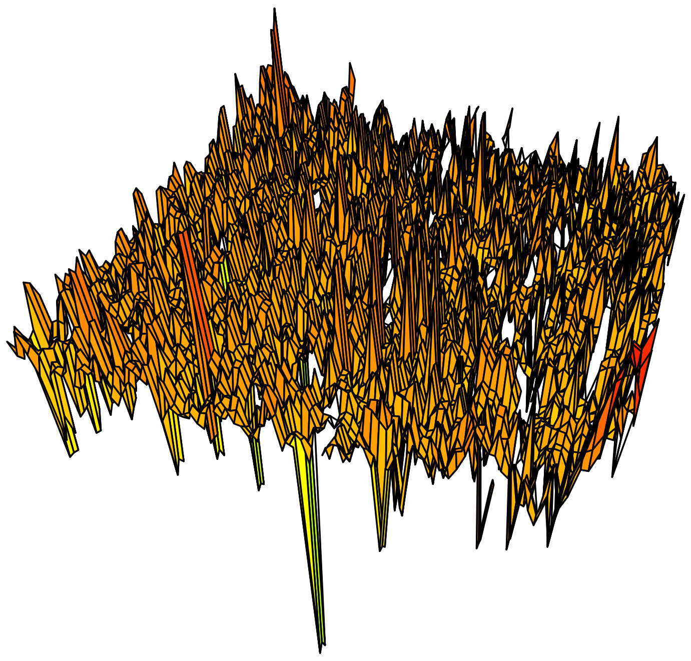

Figure 5: Perspective image of array CGH

5.2 Plot of genomic profile

```
[1] "Smoothing for each Chromosome"
[1] "Optimization of the Breakpoints"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "subset"
Time difference of 0.00030303 secs
  Region Card         Var       Mean     VarLike
1      1   82 0.008020255 0.01801656 0.008020255
3      3   46 0.011707465 0.52718028 0.011707465
[1] "aggregation"
Time difference of 0.004282951 secs
[1] "method"
[1] 7
[1] "hclust"
Time difference of 0.0006859303 secs
[1] "cluster"
Time difference of 0.009274006 secs
[1] "VERIF"
integer(0)
[1] "END VERIF"
[1] "merge"
Time difference of 0.001507044 secs
[1] "clustering"
Time difference of 0.0004398823 secs
[1] "Temps findCluster: 0.0164928436279297"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.696846e-05 secs
[1] "Temps findCluster: 0.000217914581298828"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.506111e-05 secs
[1] "Temps findCluster: 0.000215053558349609"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "subset"
Time difference of 0.000289917 secs
  Region Card         Var       Mean     VarLike
```

```
6      6  150 0.009330993 -0.0686637 0.009330993
7      7   17 0.004037443 -0.8388732 0.004037443
[1] "aggregation"
Time difference of 0.004247904 secs
[1] "method"
[1] 7
[1] "hclust"
Time difference of 0.0007021427 secs
[1] "cluster"
Time difference of 0.009041071 secs
[1] "VERIF"
integer(0)
[1] "END VERIF"
[1] "merge"
Time difference of 0.001422882 secs
[1] "clustering"
Time difference of 0.0003919601 secs
[1] "Temps findCluster: 0.0160958766937256"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 5.412102e-05 secs
[1] "Temps findCluster: 0.00177502632141113"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000221967697143555"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.506111e-05 secs
[1] "Temps findCluster: 0.000239133834838867"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 5.316734e-05 secs
[1] "Temps findCluster: 0.00185799598693848"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
```

```
[1] "clustering"
Time difference of 4.696846e-05 secs
[1] "Temps findCluster: 0.000226020812988281"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000227928161621094"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "subset"
Time difference of 0.0003120899 secs
   Region Card         Var         Mean     VarLike
20     20   34 0.005133130  0.012521000 0.005133130
21     21   12 0.003998660 -0.140525500 0.003998660
22     22   77 0.007560883  0.009708948 0.007560883
31     31   29 0.007685343 -0.107389000 0.007685343
32     32   28 0.002314400  0.058406536 0.002314400
[1] "aggregation"
Time difference of 0.004548073 secs
[1] "method"
[1] 7
[1] "hclust"
Time difference of 0.0007078648 secs
[1] "cluster"
Time difference of 0.03010607 secs
[1] "VERIF"
integer(0)
[1] "END VERIF"
[1] "merge"
Time difference of 0.001646996 secs
[1] "clustering"
Time difference of 0.0004758835 secs
[1] "Temps findCluster: 0.0377969741821289"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.696846e-05 secs
[1] "Temps findCluster: 0.000224113464355469"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
```

```
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000218152999877930"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000252962112426758"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.506111e-05 secs
[1] "Temps findCluster: 0.000216007232666016"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.506111e-05 secs
[1] "Temps findCluster: 0.000216960906982422"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000348806381225586"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 5.412102e-05 secs
[1] "Temps findCluster: 0.000236034393310547"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.696846e-05 secs
[1] "Temps findCluster: 0.000223159790039062"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
```

```
[1] "Temps findCluster: 0.000217914581298828"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.506111e-05 secs
[1] "Temps findCluster: 0.000214099884033203"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000208139419555664"
[1] "centroid"
findCluster.profileChr(profileChr = profileChr, genome = FALSE,
    lambda = lambdacluster, nmin = 1, nmax = nmax, type = type,
    param = param, verbose = verbose, method = method)
[1] "clustering"
Time difference of 4.601479e-05 secs
[1] "Temps findCluster: 0.000214815139770508"
[1] "centroid"
findCluster.profileChr(profileChr = profileCGH, region = "ZoneChr",
    genome = TRUE, lambda = lambdaclusterGen, nmin = 1, nmax = nmax,
    type = type, param = param, verbose = verbose, method = method)
[1] "subset"
Time difference of 0.0006840229 secs
   Region Card         Var         Mean     VarLike
1       1   82 0.008020255  0.018016561 0.008020255
2       2   46 0.011707465  0.527180283 0.011707465
3       3   65 0.008591739 -0.019321185 0.008591739
4       4   83 0.006330242 -0.043528434 0.006330242
5       5  150 0.009330993 -0.068663693 0.009330993
6       6   17 0.004037443 -0.838873176 0.004037443
7       7   97 0.008216950 -0.015118722 0.008216950
8       8   83 0.010461284 -0.009310735 0.010461284
9       9  174 0.008416583  0.018234207 0.008416583
10     10  151 0.007494192 -0.030153377 0.007494192
11     11  107 0.010148660 -0.032047607 0.010148660
12     12  127 0.010891078  0.014396394 0.010891078
13     13  139 0.006279640  0.020206374 0.006279640
14     14   41 0.006833627 -0.117087488 0.006833627
15     15   88 0.006625335  0.007699148 0.006625335
16     16   47 0.005636744 -0.045899702 0.005636744
17     17   71 0.011104201 -0.008812183 0.011104201
18     18   65 0.009007422  0.010588015 0.009007422
19     19   64 0.007821951  0.008468625 0.007821951
20     20   86 0.006993832  0.022920081 0.006993832
21     21   50 0.012088798  0.019128380 0.012088798
```

```
22      22   37 0.007851133  0.069814027 0.007851133
23      23   87 0.007511773  0.036766563 0.007511773
24      24   32 0.010182556 -0.074258406 0.010182556
25      25   15 0.006725160  0.056583867 0.006725160
26      26   54 0.004065299 -0.055199741 0.004065299
[1] "aggregation"
Time difference of 0.008666992 secs
[1] "method"
[1] 7
[1] "hclust"
Time difference of 0.0007891655 secs
[1] "cluster"
Time difference of 0.06020498 secs
[1] "VERIF"
integer(0)
[1] "END VERIF"
[1] "merge"
Time difference of 0.002799988 secs
[1] "clustering"
Time difference of 0.0004379749 secs
[1] "Temps findCluster: 0.0735831260681152"
[1] "Results Preparation"
```

```
> plotProfile(res, unit = 3, Bkp = TRUE, labels = FALSE, Smoothing = "Smoothing",
+     plotband = FALSE)
```

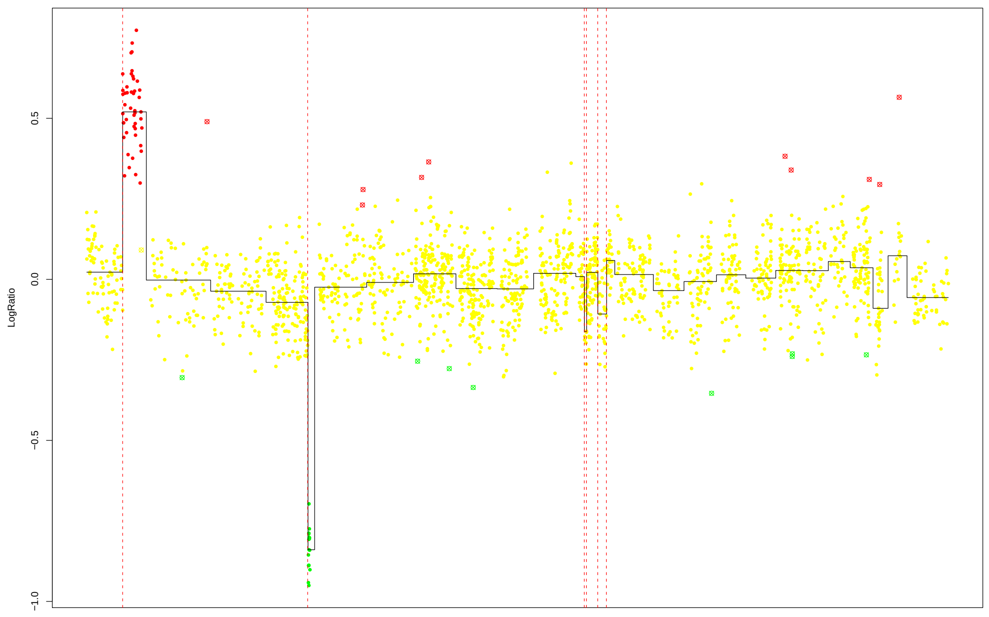

Figure 6: Genomic profile on the whole genome

```
> plotProfile(res, unit = 3, Bkp = TRUE, labels = FALSE, Smoothing = "Smoothing")
```

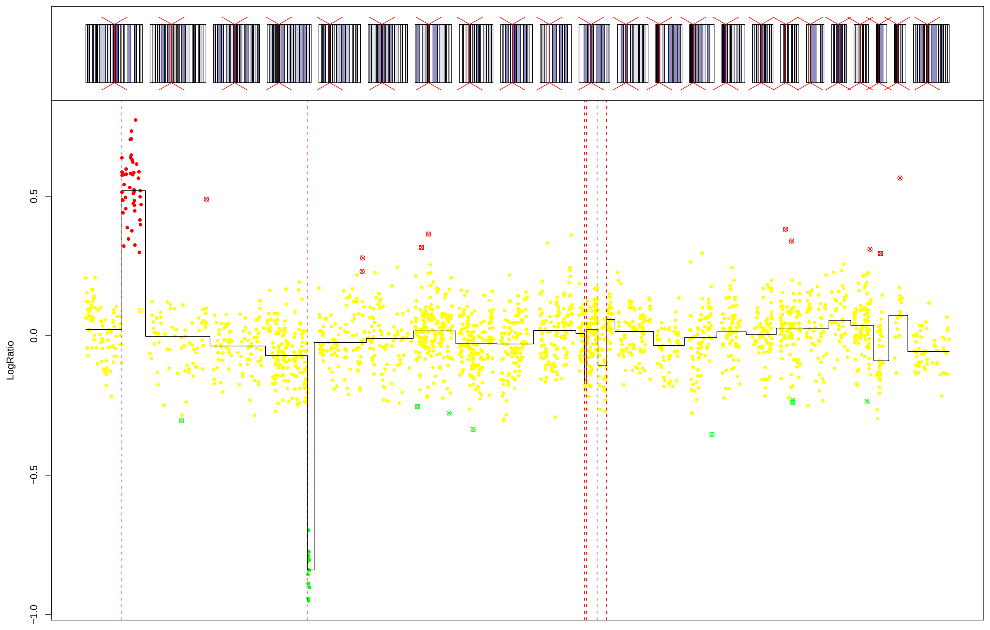

Figure 7: Genomic profile on the whole genome and cytogenetic banding

```
> text <- list(x = c(90000, 2e+05), y = c(0.15, 0.3), labels = c("NORMAL",
+     "GAIN"), cex = 2)
> plotProfile(res, unit = 3, Bkp = TRUE, labels = TRUE, Chromosome = 1,
+     Smoothing = "Smoothing", plotband = FALSE, text = text)
```

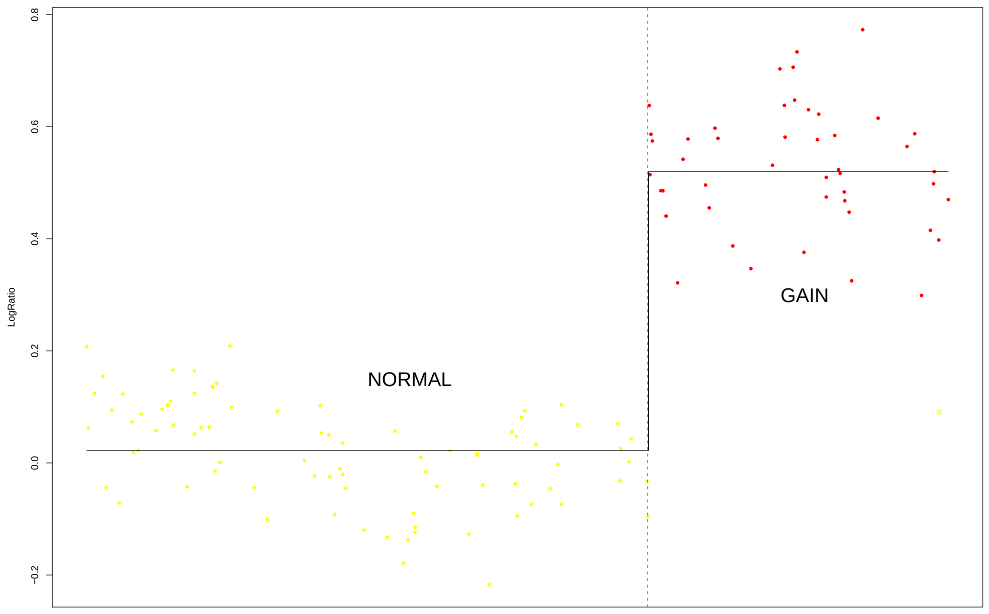

Figure 8: Genomic profile for chromosome 1

```
> text <- list(x = c(90000, 2e+05), y = c(0.15, 0.3), labels = c("NORMAL",
+     "GAIN"), cex = 2)
> plotProfile(res, unit = 3, Bkp = TRUE, labels = TRUE, Chromosome = 1,
+     Smoothing = "Smoothing", text = text, main = "Chromosome 1")
```

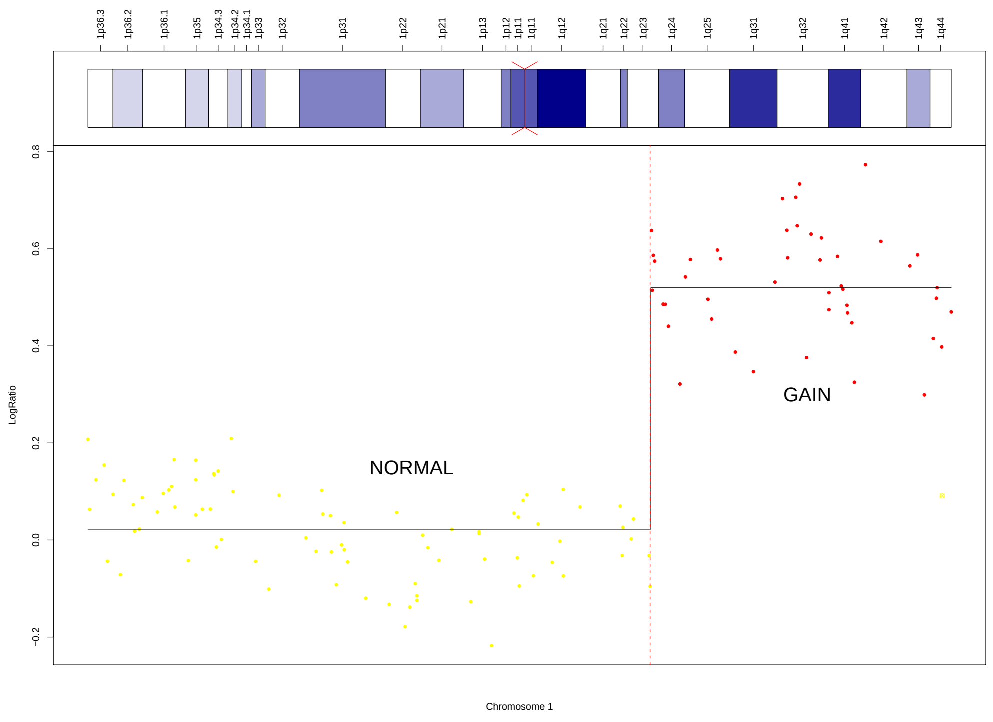

Figure 9: Genomic profile for chromosome 1 and cytogenetic banding with labels

References

Ben-Yaacov, E. and Eldar, Y. C. (2008). A fast and flexible method for the segmentation of acgh data. *Bioinformatics*, 24:i139–i145.

Billerey, C., Chopin, D., Aubriot-Lorton, M. H., Ricol, D., de Medina, S. G. D., Rhijn, B. V., Bralet, M. P., Lefrere-Belda, M. A., Lahaye, J. B., Abbou, C. C., Bonaventure, J., Zafrani, E. S., van der Kwast, T., Thiery, J. P., and Radvanyi, F. (2001). Frequent FGFR3 mutations in papillary non-invasive bladder (pTa) tumors. *Am. J. Pathol.*, 158:955–1959.

Hupé, P., Stransky, N., Thiery, J. P., Radvanyi, F., and Barillot, E. (2004). Analysis of array CGH data: from signal ratio to gain and loss of DNA regions. *Bioinformatics*, 20:3413–3422.

Ishkanian, A. S., Malloff, C. A., Watson, S. K., DeLeeuw, R. J., Chi, B., Coe, B. P., Snijders, A., Albertson, D. G., Pinkel, D., Marra, M. A., Ling, V., MacAulay, C., and Lam, W. L. (2004). A tiling resolution DNA microarray with complete coverage of the human genome. *Nat. Genet.*, 36:299–303.

Pinkel, D., Segraves, R., Sudar, D., Clark, S., Poole, I., Kowbel, D., Collins, C., Kuo, W. L., Chen, C., Zhai, Y., Dairkee, S. H., Ljung, B. M., Gray, J. W., and Albertson, D. G. (1998). High resolution analysis of dna copy number variation using comparative genomic hybridization to microarrays. *Nat. Genet.*, 20:207–211.

Polzehl, J. and Spokoiny, S. (2000). Adaptive weights smoothing with applications to image restoration. *J. Roy. Statistical Society, Series B*, pages 335–354.

Polzehl, J. and Spokoiny, S. (2002). Local likelihood modelling by adaptive weights smoothing. WIAS-Preprint 787.

Snijders, A. M., Nowak, N., Segraves, R., Blackwood, S., Brown, N., Conroy, J., Hamilton, G., Hindle, A. K., Huey, B., Kimura, K., S, S. L., Myambo, K., Palmer, J., Ylstra, B., Yue, J. P., Gray, J. W., Jain, A. N., Pinkel, D., and Albertson, D. G. (2001). Assembly of microarrays for genome-wide measurement of dna copy number. *Nat. Genet.*, 29:263–4.

Solinas-Toldo, S., Lampel, S., Stilgenbauer, S., Nickolenko, J., Benner, A., Dohner, H., Cremer, T., and Lichter, P. (1997). Matrix-based comparative genomic hybridization: Biochips to screen for genomic imbalances. *Genes Chromosomes Cancer*, 20:399–407.

Veltman, J. A., Fridlyand, J., Pejavar, S., Olshen, A. B., Korkola, J. E., DeVries, S., Carroll, P., Kuo, W.-L., Pinkel, D., Albertson, D., Cordon-Cardo, C., Jain, A. N., and Waldman, F. M. (2003). Array-based comparative genomic hybridization for genome-wide screening of DNA copy number in bladder tumors. *Cancer Res*, 63:2872–2880.